Group Collaboration for Mars Rover Mission Operations

Paul G. Backes, Kam S. Tso, Jeffrey S. Norris, Robert Steinke

Abstract— **Group collaboration capabilities have been developed for Internet-based Mars rover mission operations. Internet-based operations enables scientists to participate in daily Mars rover mission operations from their home institutions. Group collaboration enables geographically separated users to collaboratively analyze downlinked data and plan new activities for the rover. The motivation for group collaboration in Mars rover mission operations and the technologies developed to provide group collaboration are discussed. The group collaboration capability was developed for use in rover mission operations in the 2003 NASA Mars Exploration Rover mission.**

Keywords—**Rover, planning, group, collaboration, Internet.**

I. Introduction

NASA'S Mars Exploration Rover (MER) mission will land two rovers at different locations on Mars early in 2004. The rovers will be equipped with various science instruments to perform scientific analysis of the Martian terrain. Instruments will be on a mast as well as on a five degree of freedom manipulator arm. During each rover's three month primary mission, they will traverse a total of about 1 Km while exploring various locations in detail.

Command sequences will be sent to the rovers from Earth once a day and the rovers will send downlink data to Earth once a day. The mission operations system that will be used to generate the daily command sequences will be primarily located at JPL. In addition, an Internet-based science operations system will enable some scientists to participate in daily science activity plan generation from their home institutions.

Internet-based operations reduces mission costs by enabling scientists to remain at their home institutions while participating in the mission. Transportation, housing, and facilities costs are all reduced by enabling remote participation. Also, distributed operations enables participation by some scientists who would not be able to move to the JPL area during the mission.

There are various challenges to utilizing Internet-based operations for Mars rover missions. An architecture must be provided that enables numerous clients to participate simultaneously from various remote locations. Data needs to be quickly transmitted each day to the participating scientists. Group collaboration capabilities are needed to enable the various participants to work together efficiently while being geographically separated.

P. Backes, J. Norris, and R. Steinke are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California. E-mail: Paul.G.Backes@jpl.nasa.gov

K. Tso is with IA Tech Inc., Los Angeles, California. E-mail: tso@ia-tech.com

The Web Interface for Telescience (WITS) was developed to provide Internet-based distributed operations for Mars lander and rover missions. WITS was originally used as the ground operations system for the Rocky7 research rover for sequence generation [1]. An adaptation of WITS was used for Robotic Arm and Robotic Arm Camera command sequence generation for the Mars Polar Lander (MPL) mission which was to begin surface operations in December 1999 [2]. Unfortunately, communication with the MPL lander on the Martian surface was not achieved, so commanding the lander was not possible.

WITS is currently used for visualization and command sequence generation for the Field Integrated Design and Operations (FIDO) rover. The FIDO rover is a prototype rover equipped with instrumentation designed to simulate the Athena Payload for the MER mission [3], [4]. Stereo cameras on the rover's mast and body are used to image the surrounding terrain. An instrument arm places a microscopic imager on selected surface targets. An infrared point spectrometer on the mast is used to characterize the terrain. The group collaboration capability described in this paper was integrated into WITS and demonstrated for FIDO rover operations. Examples used in this paper are from FIDO rover operations.

An adaptation of WITS will be used for science planning for the MER mission. Scientists at the primary JPL operations center will collaborate with Internet-based scientists to generate the daily science activity plan.

Before the development of the group collaboration technologies described in this paper, WITS provided distributed but not collaborative participation by geographically distributed users. Downlink data was sent to remote users who could view the data, select targets, and submit subsequences to the common server. When users wanted to discuss data with colleagues, it was quite difficult to communicate to other remote users which images to view, and what features in the images were of interest. With increasing numbers of distributed users, it became very confusing when trying to communicate. Also, when a new user joined in middle of a planning session, it was difficult to get that person oriented to the current planning state.

From that distributed operations experience, the group collaboration features that were needed to make distributed collaboration efficient were identified. It was necessary to know what images and data products other users were look-

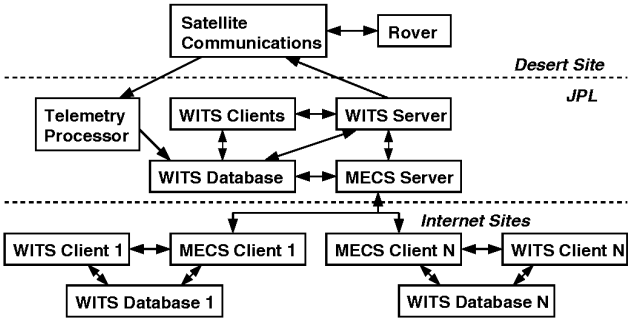

Fig. 1. FIDO Distributed Operations Architecture

ing at. It was necessary to easily be able to point out a specific terrain location in an image and have all other users see that location immediately annotated. It was needed to be able to get the attention of another user after a period of inactivity by the group. It was needed to have separate groups of people working together on specific topics. It was needed to be able to send messages to other users. The identified necessary group collaboration features were designed, developed, and integrated into WITS.

II. Distributed Rover Operations

The distributed operations architecture for FIDO rover field test operations is shown in Figure 1. There is one WITS server and many WITS clients which communicate via the server.

The process for generation of one uplink sequence begins with the downlink of data from the rover. This data is automatically processed and placed in the primary database using the Parallel Telemetry Processor (PTeP) [5]. The Multi-mission Encrypted Communication System (MECS) system distributes the new data products to the remote users. MECS was developed to provide data distribution with the necessary Internet security [6], [7].

Once users have the updated downlink data, they can view the data using the visualization views of WITS. Distributed users have the same visualization and sequence editing capabilities as users at the primary operations center. A screendump of various windows and views of WITS is shown in Figure 2. The Panorama view is a mosaic of images taken by the stereo cameras on the rover mast. The Overhead view shows the area around the rover from above. Various image types are provided with the Overhead view including color-coded elevation maps (shown in the figure) and texture maps. The 3D view shows the 3D rover and terrain. A Wedge view shows one image from a stereo camera in full size and resolution. Users can view the downlink data by themselves or collaboratively.

After viewing the downlink data, the users decide what the rover should do in the next sequence. They might do this by themselves initially, but then they join a collaborative planning meeting where 3D science targets are selected and the sequence is developed. The group features described below are used in the collaborative sequence development.

Science targets are 3D terrain locations that can be used as parameters in commands. They are shown in the views as pink circles with the name of the targets next to them. Any participating user can create a science target. An example use for a science target is as a location to point the mast-mounted spectrometer at. Target IPS-1 is used for this purpose in the example sequence. Rover waypoints are 3D terrain locations that the rover will traverse to. Waypoints are shown in the views as blue squares. Targets and waypoints entered by distributed users can be loaded, viewed, and used by all users.

The Sequence window is used to edit a command sequence. The Sequence window is a tabbed pane in the WITS Main window as shown in Figure 2. Available commands are shown in the left column and can be inserted into the sequence on the right. Various editing, simulation, and sequence verification features are provided. Users can create a new sequence and submit it to the server, edit an existing sequence, or collaboratively edit one sequence. Activities are visualized in the views when possible. In Figure 2, outlines of images in a panorama command on the cliff are shown and nine spots around the IPS-1 target are shown for the IPS scan activity.

The Resources window, shown in Figure 3, shows the resources that the rover would use if the sequence was executed. The activities are shown along the bottom and the resource plots above them. Users edit the sequence to ensure that the resources are within allocations.

III. Groups

Group collaboration capabilities were added to make collaboration with distant colleagues more efficient. Group collaboration features are provided to the user via the Group tabbed pane of the WITS Main window, as shown in Figure 4. A user can work independently viewing data products and editing sequences. When the user wants to work with other users to collaborate in viewing data products then the user joins a group.

There are two types of groups: permanent groups and temporary groups. Permanent groups are provided that have defined purposes and cannot be deleted. For example, in Figure 4 the sowg group represents the permanent Science Operations Working Group which includes all mission scientists. Temporary groups can be created and deleted by users.

The Group window “Group” menu is used for creating new groups, deleting groups, joining a group, and leaving a group. A group is created by selecting the “New” menu item. The user gives the group a name and then the new group is available to all users to join. A group is deleted by selecting the “Delete” menu item, and then selecting the

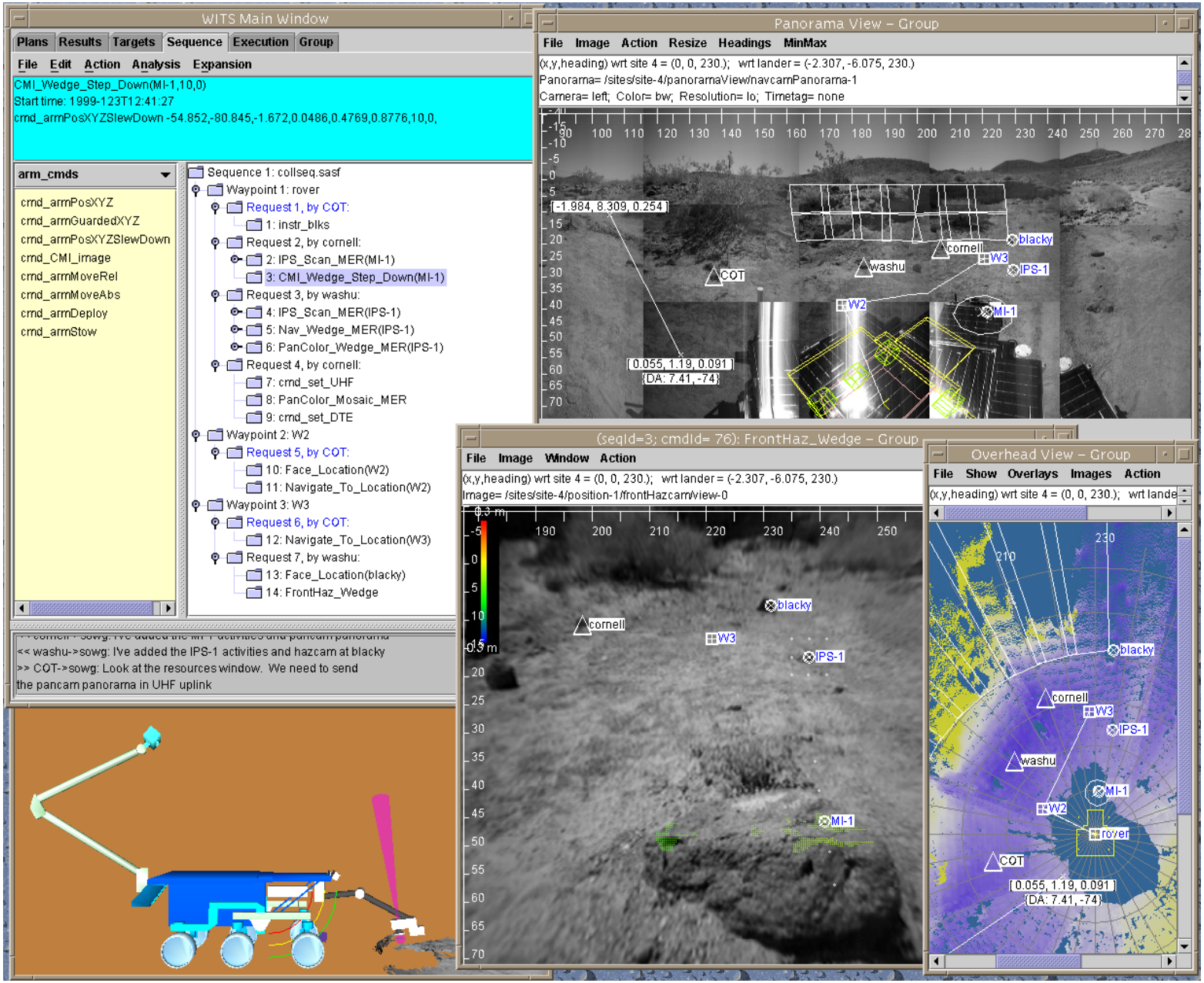

Fig. 2. WITS Sequence Window and Panorama, Overhead, 3D, and Wedge Views.

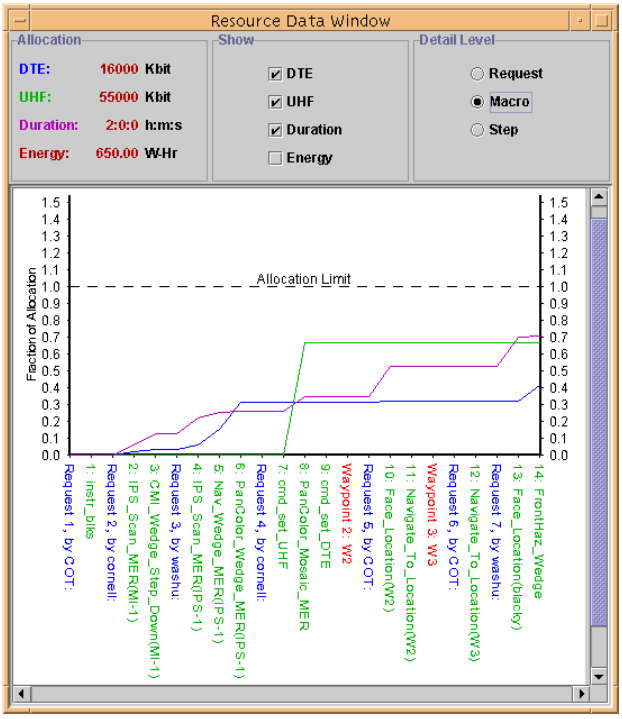

Fig. 3. Resources Window

group name. A group cannot be deleted if it is a permanent group or if there are any current users in the group.

A user joins a group by selecting “Join” in the Group window “Group” menu and then selecting the available group name. A user leaves the group they are currently in by selecting the “Leave” menu item.

All users who are currently using WITS are listed in the left column of the Group window, as shown in Figure 4. The group that a user is in is shown in square brackets to the right of the user name, if the user has joined a group.

IV. Group Views

To ensure that all group members are viewing the same information, the group view feature was implemented. A view is turned into a group view by selecting the “Add To Group” item in the File menu of the Panorama, Overhead, Wedge, and 3D views. When a view becomes a group view, it is opened in all group members’ clients and the word Group is added to its title.

The “View” menu of the Group window provides menu items for management of the group views. The “Refresh Group Views” menu item opens all views of the user’s current group that are not opened, for example if the user closed one or more group views for some reason but now wants to make sure all group views are open. The “Close Non-Group Views” menu item closes all views except those in the current group. The “Close Other-Group Views” menu item closes all group views that are not in the current group. This is useful for when a user changes groups and wants to close the views from the previous group. By design, the views of the previous group are not automatically closed when a user leaves a group or joins a new group. In the above menu items, Close means to close the group view, but leave it as an item in the group state on the server. The “Delete All Group Views” menu item deletes all views of the current group from all group members’ clients. This is only done by the group administrator. Delete means that the view is deleted from the group state on the server as well as closed as a view on the clients.

V. Group Markers

The Group window “Marker” menu is used to manage group markers. A marker is a 3D location selected by a group user and displayed on all other group users’ client views. Examples are shown in Figure 2. To create a marker, a user first selects a pixel in an image in a Panorama or Wedge view. The 3D coordinates are then displayed. This location is converted to a marker by selecting “Add” in the “Marker” menu. A green triangle is drawn at the location in all Panorama, Wedge, and Overhead views of all group members and the name of the user who created the target is displayed next to the marker.

The “Delete” menu item of the “Marker” menu deletes the user’s marker in all group member clients’ views. The “Delete All” menu item deletes all markers from all users in all group member clients’ views. The “Show” checkbox menu item allows the user to turn on and off display of group markers.

VI. Group Messages

Group messages are used for textual communication between group members. It is assumed that during collaboration, telephone-based teleconferencing or Internet telephony are used for verbal communication. All received group messages are displayed in the Messages area of the Group window. To send a group message, a user types the message in the “Inputs” area of the Group window and then selects the “Send” button. Who the message is sent to and how it is sent are controlled by the selection in the menu below the Send button. If “Personal” is selected, then the message will only be sent to the user that has been selected in the users list in the left column of the Group window. The selected user does not have to be in the group that the current user is in. If “Group” is selected, then the message will be sent to all members of the current group. If “Announcement” is selected, then the message will be displayed in a pop-up window in all group members’ clients. The “Send Beep” checkbox of the Group window indicates whether the user wants a beep to be sounded on all group members’ clients when their message is received.

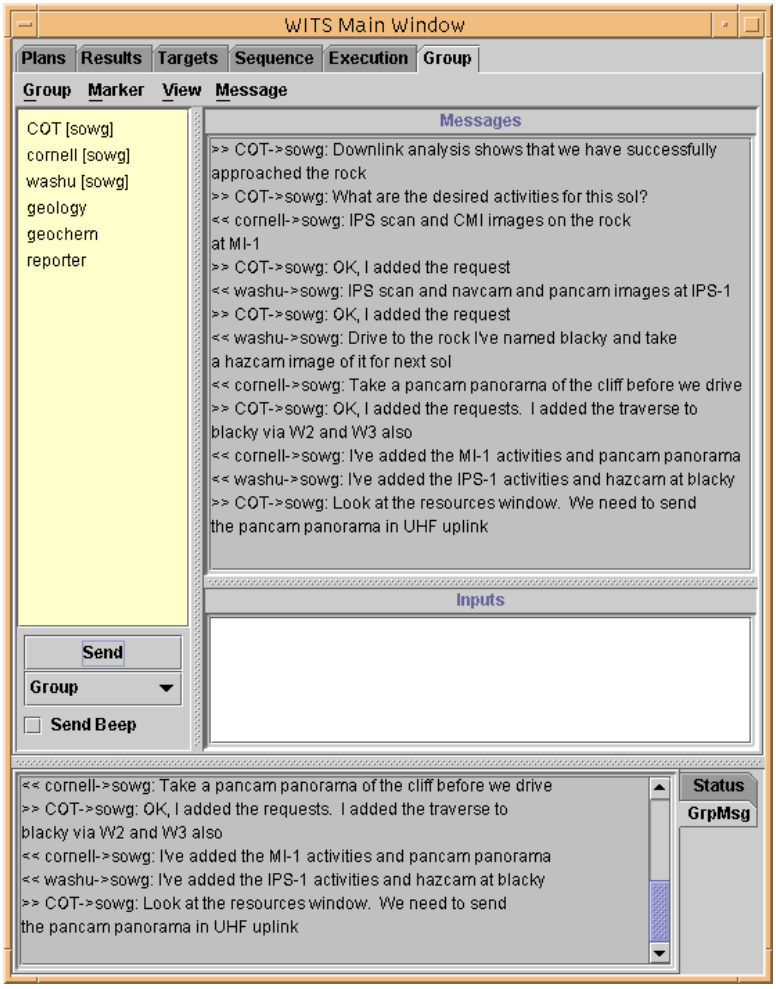

Fig. 4. Group Window

VII. Group State and Initialization

The server maintains a state for each group. The state includes information on all current client users, all current markers, cumulative group messages, and current group views. When a user joins a group, their client is automatically initialized with the group state, i.e., all the group views are automatically opened, all the markers are displayed, and the group messages are displayed. The group state is also useful for on-going group users. If a group user deletes one of more group views for some reason, then they can automatically reopen them by selecting "Refresh Group Views" in the View menu of the Group window, as discussed above. This is possible due to the server maintaining the group state.

VIII. Collaborative Sequence Editing

Collaborative sequence editing enables users to simultaneously edit a common sequence. This reduces the sequence generation time by having different parts of the sequence developed in parallel.

The basic hierarchy of a sequence is sequence, request, activity, command. Any number of requests can be in a sequence, any number of activities can be in a request, and any number of commands can be in an activity. Resource allocations and time constraints are specified at the request level. Resource allocations include time duration, energy, and data volume. Time constraints specify specific times when tasks must begin or end by.

For collaborative sequence editing, ownership of the sequence is specified at the sequence and request levels. The owner of a sequence can modify the structure of the sequence including adding and deleting requests and specifying owners for each request. The sequence owner cannot modify the activities inside a request that is owned by a different user (although the sequence owner can delete a complete request). The owner of a request can add, delete, and modify the activities inside the request, but cannot modify the constraints and resource allocations for the request.

To collaboratively edit a sequence, a user first checks out the sequence. The sequence is copied from the common

server to the user's WITS client. The user then modifies the sequence as appropriate for their ownership. Updating the common server sequence with the user changes is called merging.

To merge a sequence owner's changes to the common sequence at the server, the sequence owner user selects a menu item to modify the sequence structure. The original server sequence is copied and given a version number. Then the structure changes are made to the sequence. All requests in the modified sequence are used in the new sequence. For all requests that were in the original sequence, the activities are copied from the original sequence and placed in the new sequence.

A request owner merges changes to the common server by selecting a menu item to merge their changes to the common server. The original server sequence is copied and given a version number. Then the requests in the original sequence that the current user is owner of are updated with the activities in the modified sequence.

IX. Implementation

WITS implements a client/server architecture to support group communications and other mission planning and operations activities. The WITS server is implemented as a Java application running on a computer at JPL. The WITS client is downloaded from the JPL WITS website and run as a Java application.

RMI (Remote Method Invocation) is used for client/server communications since WITS is developed in a pure Java environment and RMI provides a number of advantages. RMI can pass full objects as arguments and return values, not just predefined data types. This means that complex types, such as a standard Java hashtable object, can be passed as a single argument. RMI provides secure channels between client and server by using the secure socket factory. RMI provides a means for clients running behind a firewall to communicate with remote servers via tunneling messages through HTTP POST calls.

The WITS server implements a group server RMI object to provide group collaboration services to the clients. When the user of the WITS client logs in, the client obtains the reference of the group server object from the RMI registry running at the WITS server site. The client then invokes the `userLogin` method with the username and password arguments for login validation. At the same time, the WITS client passes its group client object to the server. This enables the server to send messages to the client using a technique commonly known as server-to-client callback.

Group collaboration in WITS uses both synchronous and asynchronous remote method calls. Synchronous calls can provide immediate results to the clients. For example, in the `userLogin` call, the user can know if a wrong password has been entered. However, synchronous calls can hang the client for an indefinite length of time if the server or network is overloaded. Thus, to improve both user experience and system performance, we implement most remote method calls to be asynchronous so that the client and server are loosely coupled. With asynchronous communication, the server can schedule its processing group collaboration requests more efficiently because it does not have to reply to each call immediately. For the client, it can continue to handle other tasks such as screen update and user input after it submits a group collaboration request.

The group server maintains a queue of pending group collaboration requests submitted by the clients. The requests are processed in the first-come-first-serve basis. As multiple clients can concurrently submit asynchronous requests and make synchronous remote method calls, attention must be paid to ensure that group state data, such as the list of users, will be updated atomically and without causing deadlock. The Java language provides the *synchronized* primitive that automatically guarantees that two or more threads will not be updating the object or executing the method that has been declared as synchronized. We make use of this feature to implement the serialization of updating the group state data.

The states of group collaboration are always maintained at the Group Server. They include the lists of users and groups, group messages, markers, and views. When a user joins a group, the client will be updated with the current state of the newly joined group. This ensures the client to have the same group views, group markers, and discussions among group members. The group messages are also logged to a file for creating a record of the discussions.

Another function of the group server is to detect terminated clients that have not properly logged out. It sends periodic probe messages to each client which will cause the remote method call to fail if the client no longer exists. When all the clients are logged out, the group server cleans up the states and closes the group message log file.

WITS utilizes APIs (application programming interfaces) provided by the Java 2 platform. The Swing graphical user interface components, such as tabbed pane, split pane, tree view, etc., are used to implement the user control and viewing functions. Java 2D is used to manipulate images and draw 2D objects, while Java 3D is used to render solid models and Mars terrain.

X. Future Plans

The goal of future work is to make distributed collaboration even more convenient. The planned environment requires support for two fundamentally different kinds of communication. First, remote users must have access to data products. Second, groups of users must be able to collaborate. In the first case, availability is a large concern. The communication system should be resilient to lost messages and periods of disconnection. The user should be able to cache copies of data products locally and continue

to work in disconnected mode. To provide availability and disconnected operation, an appropriate technology is a gossip protocol [8], [9], [10].

The second case is concerned with synchronous collaboration where users are present at the same time, but perhaps not at the same location. A primary concern for synchronous collaboration is the creation of shared context, e.g., if all collaborators are viewing the same data product they can refer to features in that data product more easily. Disconnected operation is not a concern for this type of communication because disconnection completely eliminates the possibility of synchronous collaboration. To provide shared context in a connected environment an appropriate technology is a group multicast protocol [10], [11], [12]. In the future we plan to support both modes of communication, and explore interactions that occur when collaborators have both methods available.

A gossip protocol functions in the following manner. The nominal state of this system is a disconnected state. Occasionally, two sites connect and exchange information about which messages each has received. Then each site forwards any messages not yet received by the other. A downlinked data product would be placed in the gossip log and eventually reach all sites. If the data product itself is too big a message announcing its availability can be sent. The system doesn't have to worry about detecting and repairing broken communication links or resending messages because disconnection and resending are part of the nominal operation of gossip.

A group multicast protocol provides efficient delivery of messages to a set of destinations. An application can choose whether messages are delivered in a reliable or unreliable manner. In addition, ordering conditions can be enforced on messages such as total order, causal order, or FIFO order. A group collaborator may take an action such as opening a new data product or making annotations, and that action should be visible to all other group members. To achieve this, the action is described in a message that is delivered by a group multicast protocol enforcing reliability and total ordering. This way, every member of the group sees the same set of actions in the same order so their shared context remains consistent.

XI. Conclusions

Daily participation in Mars rover mission operations by geographically distributed scientist users via the Internet will greatly benefit Mars missions both in cost savings and increased participation by scientific experts. The new group collaboration features described in this paper enable geographically distributed users to collaborate effectively in planning daily rover mission activities.

Acknowledgments

The work described in this paper was funded by the TMOD Technology Program, the Mars Exploration Technology Program, and the SBIR Program, and performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] S. Hayati, R. Volpe, P. Backes, J. Balaram, R. Welch, R. Ivlev, G. Tharp, S. Peters, T. Ohm, and R. Petras, "The Rocky7 rover: A Mars sciencecraft prototype," in *Proceedings IEEE International Conference on Robotics and Automation*, Albuquerque, New Mexico, April 1997, pp. 2458–2464.

[2] Paul G. Backes, Kam S. Tso, Jeffrey S. Norris, Gregory K. Tharp, Jeffrey T. Slostad, Robert G. Bonitz, and Khaled S. Ali, "Internet-based operations for the mars polar lander mission," in *Proceedings IEEE International Conference on Robotics and Automation*, San Francisco, California, April 2000, pp. 2025–2032.

[3] R. Arvidson, Paul Backes, E. Baumgartner, D. Blaney, L. Dorsky, A. Haldemann, R. Lindemann, P. Schenker, and S. Squyers, "Fido: Field-test rover for 2003 and 2005 mars sample return missions," in *30th Lunar and Planetary Science Conference*, Houston, Texas, March 15–19 1999.

[4] P.S. Schenker, E.T. Baumgartner, L.I. Dorsky, P.G. Backes, H. Aghazarian, J.S. Norris, T.L. Huntsberger, Y. Cheng, A. Trebi-Ollennu, M.S. Garrett, B.A. Kennedy, and A.J. Ganino, "Fido: a field integrated design & operations rover for mars surface exploration," in *Proceedings 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-'01)*, Montreal, Canada, June 18-21 2001.

[5] Jeffrey S. Norris, Paul G. Backes, and Eric T. Baumgartner, "PTeP: The parallel telemetry processor," in *Proceedings IEEE Aerospace Conference*, Big Sky, Montana, March 2001.

[6] Jeffrey S. Norris and Paul G. Backes, "Wedds: The WITS encrypted data delivery system," in *Proceedings IEEE Aerospace Conference*, Big Sky, Montana, March 2000.

[7] Robert Steinke, Paul G. Backes, and Jeffrey S. Norris, "Distributed mission operations with the multi-mission encrypted communication system," in *Proceedings IEEE Aerospace Conference*, Big Sky, Montana, March 9-16 2002.

[8] M.J. Lin, K. Marzullo, and S. Masini, "Gossip versus deterministically constrained flooding on small networks," in *Proceedings of the 14th International Conference on Distributed Computing (DISC 2000) a.k.a Lecture Notes in Computer Science*, 2000, vol. 1914, pp. 253–267.

[9] G. Fertin, "Hierarchical broadcast and gossip networks," *Information Processing Letters*, vol. 73, no. 3-4, pp. 131–136, February 29 2000.

[10] S.M. Hedetniemi, S.T. Hedetniemi, and A.L. Liestman, "A survey of gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, pp. 319–349, Winter 1988.

[11] Y. Amir, C. Danilov, and J. Stanton, "A low latency loss tolerant architecture and protocol for wide area group communication," in *Proceedings of the International Conference on Dependable Systems and Networks*, 2000, pp. 327–336.

[12] I. Rhee, S.Y. Cheung, P.W. Hutto, A.T. Krantz, and V.S. Sunderam, "Group communication support for distributed collaboration systems," *Cluster Computing*, vol. 2, no. 1, pp. 3–16, 1999.